

# FISF: Better User Experience using Smaller Bandwidth for Panoramic Virtual Reality Video

Lun Wang<sup>†</sup>, Damai Dai<sup>†</sup>, Jie Jiang<sup>†</sup>, Tong Yang<sup>†</sup>, Xiaoke Jiang<sup>\*</sup>, Zekun Cai<sup>†</sup>, Yang Li<sup>\*</sup>, Xiaoming Li<sup>†</sup>

<sup>†</sup> Peking University, <sup>\*</sup> Kandao Technology Co., Ltd.

## ABSTRACT

The panoramic video is widely used to build virtual reality (VR) and is expected to be one of the next generation Killer-Apps. Transmitting panoramic VR videos is a challenging task because of two problems: 1) panoramic VR videos are typically much larger than normal videos but they need to be transmitted with limited bandwidth in mobile networks. 2) high-resolution and fluent views should be provided to guarantee a superior user experience and avoid side-effects such as dizziness and nausea. To address these two problems, we propose a novel interactive streaming technology, namely Focus-based Interactive Streaming Framework (FISF). FISF consists of three parts: 1) we use the classic clustering algorithm DBSCAN to analyze real user data for Video Focus Detection (VFD); 2) we propose a Focus-based Interactive Streaming Technology (FIST), including a static version and a dynamic version; 3) we propose two optimization methods: focus merging and prefetch strategy. Experimental results show that FISF significantly outperforms the state-of-the-art. The paper is submitted to Sigcomm 2017, VR/AR Network on 31 Mar 2017 at 10:44:04am EDT.

## 1. INTRODUCTION

### 1.1 Background and Motivation

The panoramic video is widely used to build virtual reality (VR) and is expected to be one of the next generation Killer-Apps. It is widely used in online multimedia [5], video surveillance [12] and robotics [14] applications because of its good interactivity.

As shown in Figure 1, a panoramic VR video is typically a two-dimensional rectangular video. Video players will map it onto a mesh (typically, sphere or skybox [8]), and render it on users' screen like helmet-mounted devices (HMD) or mobile phones. When watching a panoramic VR video, the users can navigate the scenes interactively by changing their viewpoints and viewing directions. One point in a video is described by a quadruple:  $(t, x, y, z)$ .  $t$  is the timing of the video,  $(x, y)$  are the spatial coordinates, and  $z$  is the users' viewing directions.

Transmitting panoramic VR videos is a challenging task because of two problems: 1) panoramic VR videos are typically much larger than normal videos (fluent

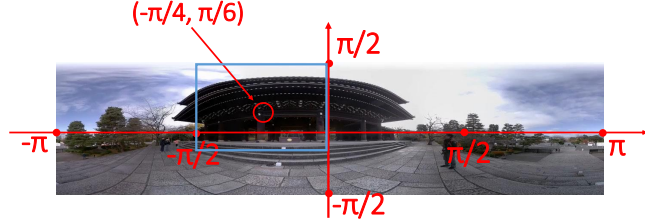


Figure 1: Illustration of Coordinate.

transmission requires 10+Mbps) but they need to be transmitted with limited bandwidth in mobile networks. 2) high-resolution views should be provided to guarantee a superior user experience and avoid side-effects such as dizziness and nausea.

### 1.2 Limitations of Prior Art

The naive solution directly transmits  $360^\circ \times 180^\circ$  high-resolution panoramic VR videos, which consumes much bandwidth and causes lag phases, where lag phase means that the users' screens keep unchanged or vague for seconds. Facebook provides a solution called interactive streaming technology [1]. It produces 32 copies of videos, each of which contains only a fixed high-resolution area (*e.g.*,  $90^\circ \times 120^\circ$  scene, see the blue rectangle in Figure 1), while the other areas are low-resolution. It chooses the most appropriate copy and transmits it to users based on the users' current viewpoints. In this way, the bandwidth for transmission is significantly reduced. However, when the users' viewpoints change, a new copy needs to be transmitted. Transmission latency inevitably incurs lag phases on the users' screens. The lag phases, caused by the transmission latency (typically for seconds), will significantly degrade the user experience because the users can only watch low-resolution panoramic VR videos during lag phases. Because of the method's large influence and practicability, we consider it as the state-of-the-art. Another solution is proposed by Ochi Daisuke and several other researchers [11], which is similar to Facebook's solution. There are several other cutting-edge researches aiming at addressing the bandwidth and user experience issues, such as object-based transcoding [13], perception-based scheduling [10], and active video anno-

tations [6, 7]. They are based on different technologies, such as object detection, annotation, *etc.* Our proposed solution is different from these researches because it is based on analysis of real-world user-watching traces.

### 1.3 Our Proposed Solutions

In this paper, we propose a **focus-based interactive streaming framework (FISF)**. FISF consists of a **video focus detection (VFD)** algorithm based on user data analysis, a static and a dynamic **focus-based interactive streaming technologies (FIST)**, and two further optimizations: focus merging and prefetch strategy. FISF achieves a much smaller number of lag phases, and makes users enjoy high-resolution views with low bandwidth.

FISF is based on our key observation from the tests and analysis of real panoramic videos: when the users watch a panoramic video, there are some viewpoints more likely to be watched for a long time, namely **focuses**. The framework of FISF is shown in Figure 2. First, we propose a VFD algorithm to detect video focuses, where VFD leverages the well-known DBSCAN [9] algorithm. Second, similar to the state-of-the-art [1], we also produce multiple video copies, and each copy also contains a small area of high-resolution video. In the state-of-the-art solution, the high-resolution area is irrelevant to the video focuses. While in our solution, the high-resolution area of each copy contains one or more video focuses. The state-of-the-art [1] saves bandwidth at a high cost of large switching number and long lag phase time. In contrast, FISF reduces switching number and lag phase time, and even saves more bandwidth.

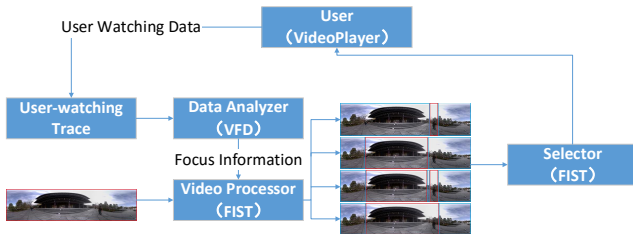


Figure 2: Hierachy of FISF.

### 1.4 Key Contributions

- We propose the idea of detecting video focuses by analyzing user data, and use it to improve panoramic VR video transmission. We also propose a framework to augment video transmission with video focus detection.
- We present a concrete algorithm for data-based video focus detection, two versions of focus-based interactive streaming technologies: a static version and a dynamic version, and two further optimizations.
- We simulate our framework and perform extensive experiments using real user-watching traces to evalu-

ate the improvement in terms of user experience and bandwidth.

## 2. METHODOLOGY

In this section, we present the three parts of FISF. First, we present a video focus detection method, called **video focus detection based on user data analysis (VFD)**, which uses DBSCAN clustering algorithm [9]. Second, to improve the user experience and save bandwidth for VR videos, we propose an algorithm, namely **Focus-based Interactive Streaming Technology (FIST)** with a static version and a dynamic version. Third, we propose further optimization methods, including focus merging and prefetch strategy.

### 2.1 Part I: VFD

In this subsection, we present the first part of FISF, namely VFD. It serves to detect the focuses of videos, and provides the focus information, so as to help the video processor produce different copies of videos. As there is no algorithm to detect VR video focuses based on user data, we tried several classic clustering algorithms, and (DBSCAN) [9] exhibits the best performance. According to experimental results on real user data, the focuses detected by VFD highly conform with empirical results and serve well for FIST.

**Rationale:** Our key observation is that users tend to focus on only some specific points, and ignore other parts when watching a video, especially panoramic VR video, because only part of the video can be seen. The intuition is further confirmed by a simple analysis on real user data. For example, when users look at the picture shown below, the empirical probability distribution of attention is shown in Figure 3. Those maximum points with the highest probability to be focused on are focuses. There are two approaches to achieving focus detection: 1) based on computing graphics and 2) based on data analysis. In this paper, we choose the second solution because of two reasons: 1) the second solution is more accurate than the first one because it is based on real user-watching traces; 2) the second solution is more time-efficient in terms of algorithm complexity.

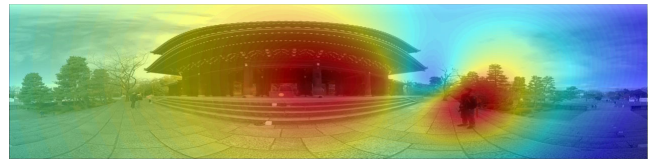


Figure 3: Probability Distribution of Attention. Red area means high attention probability. Blue area means low attention probability.

**DBSCAN:** VFD chooses DBSCAN to detect focuses. The DBSCAN algorithm views clusters as areas of high density separated by areas of low density, where density is defined by two parameters **min-samples** and **eps** [9].

The algorithm examines every sample and find its neighbors (which means samples within a distance of  $\epsilon$ ). If the number of neighbors is larger than  $\text{min-samples}$ , we say the area near the determined sample is dense and call the sample **core sample**. If a sample is not a core sample but it is a neighbor of a core sample, we still put it in the cluster. However, if a non-core sample does not have any core sample neighbors, it is not part of any cluster. The key challenge using DBSCAN for focus detection is the selection for parameters. We address this problem using a validation part in VFD.

**VFD:** Figure 4 shows the flow chart of VFD and Table 1 shows the user-watching trace format. The VFD algorithm is composed of the following three steps. 1) Data filtering. Data filtering preprocesses the user-watching traces and filters out the “dirty data” such as a long-time lag phase without any interactive behavior. Then the “clean data” are divided into two sets: a training set and a validation set. 2) Clustering. DBSCAN is applied on the training set with preset parameters to provide “preliminary”. 3) Validation. These focuses, combined with the validation set, are used to simulate the real user behaviors. This will produce feedback to the DBSCAN algorithm and new parameters will be chosen according to the feedback. This procedure will stop until convergence or it reaches the preset upper bound of iterations.

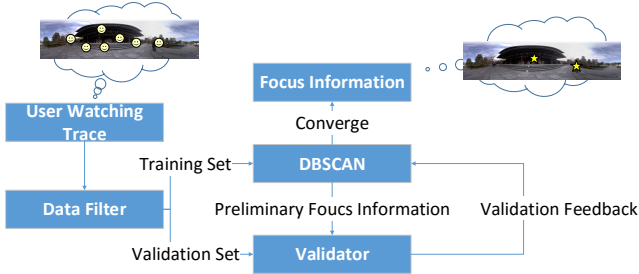


Figure 4: Workflow of VFD.

ID	Timing	x	y	z
101	0.329	-1.542456	-0.2082523	0.2079071
101	1.328	-1.54239	-0.2015937	0.2011556
102	0.045	-1.495437	-0.02360264	1.607887
101	2.336	-1.541883	-0.198082	0.1975058
...	...	...	...	...

Table 1: Watching Record Data Table.

The finite state machine (FSM) of DBSCAN and validation is shown in Figure 5. The initial parameters for DBSCAN are preset, so focuses detected are likely to be not accurate, and thus have a poor performance in saving bandwidth and improving the user experience. To address this issue, we set a validation part to verify the performance by simulating real user behavior

using validation set. If the performance is better, we change the parameters in the same direction with a fixed step length (if this is the first time of validation, choose the direction randomly). Otherwise, we change the parameters in the reverse direction with a reduced step length. The procedure stops either when it converges or when the number of iterations reaches the preset upper bound. In section 3, we carry out an experiment to decide the appropriate preset parameters (see Figure 8).

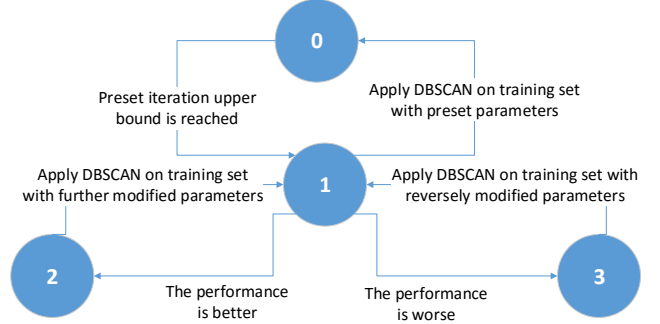


Figure 5: FSM of VFD.

## 2.2 Part II: FIST

In this subsection, we propose two versions of FIST. We first introduce the basic framework for FIST. Then we introduce two versions of FIST: a static version and a dynamic version, suitable for different videos.

**Basic FIST Framework:** The basic framework of FIST is shown in Figure 6. FIST consists of two main parts: 1) A video processor used to produce different copies of videos; 2) A selector to choose which copy to transmit according to the current user viewpoint. Focus information from VFD is passed to the video processor, and several copies are produced. Each copy, namely **fcopy** has a high-resolution area covering one or more focuses while other parts are low-resolution. When users watch videos, watching devices like helmet-mounted devices or mobiles phones will detect the users’ viewpoints and report them to the selector. The selector will choose the copy with the corresponding video to transmit. Note that the producer also produces four copies, namely **bcopy**, each with a  $90^\circ$  high resolution area and together covering the whole video. When the users’ viewpoints are out of any focus, the selector will transmit one of these four copies according to the viewpoints.

This method may introduce extra time and space consumption, but they are both ignorable compared to the bandwidth bonus. First, the preprocessing needs to be done only once. Second, the low-resolution area of a video consumes much smaller memory space than the high-resolution area, so the copies will just consume a little extra memory than a  $360^\circ \times 180^\circ$  high-resolution

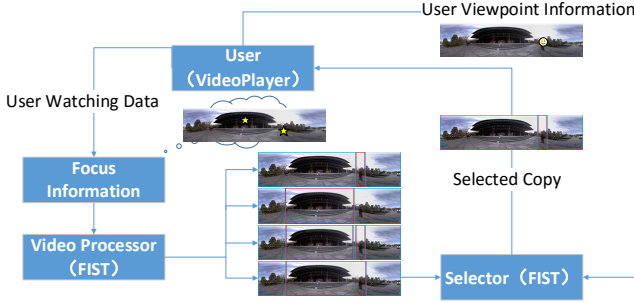


Figure 6: Framework of FIST.

video. Third, storage is cheap for panoramic VR video provider so memory usage is also not a problem.

**Static FIST:** Static FIST uses static focus information to produce video copies. To provide focus information, VFD ignores the time dimension ( $t$ ), and applies DBSCAN on  $x$  and  $y$  dimension, thus the focuses only contain spatial information. The strategy of selecting which copy to transmit is shown in Algorithm 1.

**Dynamic FIST:** Dynamic FIST is based on the key observation that focuses move in a predictable pattern in many videos. For example, in a broadcasting video for a basketball game, the focus is likely to follow the ball. If we apply static FIST to this video, it will switch the copies transmitted back and forth, leading to frequent lag phases. Dynamic FIST addresses this problem by applying DBSCAN on  $x$ ,  $y$ , and  $t$  dimensions. Therefore, the focuses contain time information. When we preprocess the videos to produce copies of the original videos, the copy covering dynamic focuses will have a moving high-resolution area. Note that the implementation of Dynamic FIST is still our undergoing work. The selection algorithm for dynamic FIST is shown in algorithm 1.

**Static Version vs. Dynamic Version:** Note that these two versions adapt to different situations. When a video has static focuses, static version will definitely have better performance due to more accurate focuses. However, when the focus moves, dynamic version will be better because in static version, the server will need to switch copies frequently and introduce many lag phases.

### 2.3 Part III: Undergoing Work

In this section, we propose two optimization approaches: focus merging and prefetch strategy.

**Focus Merging:** We observe that users sometimes change frequently between two near focuses. Although one focus may be covered by a copy aiming at covering another focus, the marginal part of users' view may be vague. To address this issue, we produce a copy covering these two near focuses to prevent this problem. Figure 7 shows focus merging technology.

**Prefetch Strategy:** The second optimization is prefetch strategy. One can construct a probabilistic

---

#### Algorithm 1: Copy Selection for FIST

---

**Input:** User viewpoint:  $V_u$

Current copy being transmitted:  $C_c$

**Output:** Selected Copy

```

1 if  $V_u$  in the high-resolution area of  $C_c$  then
2   | Keep transmitting  $C_c$ 
3   | goto Done
4 else
5   | # Dynamic version
6   |  $S_n$  = find dynamic fcopies containing  $V_u$ 
7   | # Static version
8   |  $S_n$  = find static fcopies containing  $V_u$ 
9   | if  $|S_n| == 0$  then
10  |   |  $C_n$  = find bcopies containing  $V_u$ 
11  |   | else
12  |   |   |  $C_n$  = randomly choose a fcopy from  $S_n$ 
13  |   |   | Switch to transmit  $C_n$ 
14 Done

```

---

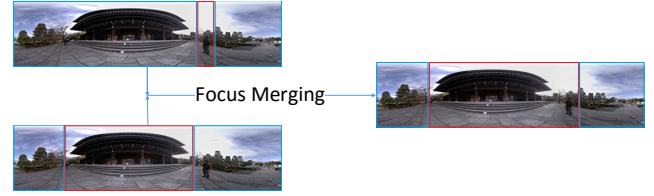


Figure 7: A Focus Merging Example: Red part means high-resolution area, blue the opposite.

model from user data and apply it to predict the movement of viewpoint. Based on its prediction of viewpoint, the server and the client will both leave some bandwidth to prefetch the predicted copies. If the users behave as predicted, they will immediately see the high-resolution part without any lag phase.

## 3. EXPERIMENTAL EVALUATION

### 3.1 Experimental Setup

**User-watching Traces:** We use real user-watching trace collected by Kandao Technology Co., Ltd. [3] to simulate the runtime bandwidth and flow switching behaviors. To objectively show the performance of FIST and the state-of-the-art, we select five different kinds of VR videos to carry out experiments. These five videos are available at the website [4]. The first video is a grouping dancing, which has multiple static focuses. The second is a video of constructing a bridge, which has a moving focus. The third is a VR broadcast, which has only one static focus. The fourth is a travel advertisement, which has no obvious focus. The fifth is a solo dance, which has a static focus. We have released our source codes at GitHub [2].



**Computer Setting:** We run simulation experiments on a HP OMEN Notebook PC 15 with 8 CPU cores and 16 GB memory.

**Metrics:** We define four metrics to evaluate the transmission performance.

- **Switching number:** defined as the number of switching among copies.
- **Standstill Time:** defined as the lasting time of lag phases when the video is standstill.
- **High quality rate:** defined as  $T_h/T_t$ , where  $T_h$  denotes the time during which the users watch high-resolution areas, and  $T_t$  denotes the total time of the watching trace.
- $\alpha$ :  $\alpha$  is defined as  $n/N$ , where  $n$  denotes the number of bytes transmitted using different algorithm, and  $N$  denotes the number of bytes when transmitting the  $360^\circ \times 180^\circ$  high-resolution videos.

### 3.2 Experimental Results

**Parameter selection:** As mentioned above, DBSCAN has two main parameters: **eps** and **min-samples**. The choice of these two parameters will significantly affect the accuracy of focus detection and the performance of FISF. Figure 8 shows the relation between focus number and eps. We can see that for each video, the focus number declines with the increase of eps. Our experiment also shows that the results do not have a clear relation to min-samples, thus we omit the corresponding experimental results. According to the experimental results, we preset eps as 0.3, min-samples as 100 for static FIST and eps as 0.2, min-samples as 30 for dynamic FIST.

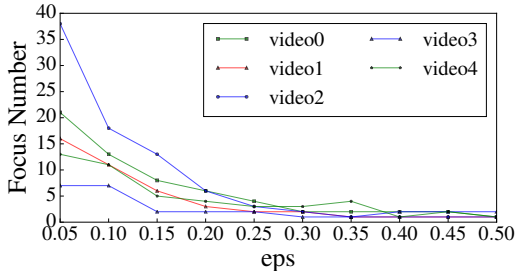


Figure 8: Focus Number vs. eps.

**Switching Number vs. Video ID :** Our results show that *our static version can reduce the number of copy switching by [23.4%, 49.1%], with a mean of 37.3%, and our dynamic version can reduce that by [15.3%, 41.7%], with a mean of 28.3%*, compared to the classic algorithm. As shown in Figure 9, the x axis represents the video ID and y axis represents the number of copy switching.

Copy switching may cause additional lag phases and computation cost. Thus the switching number should be reduced as much as possible. For most videos, the number of focuses is usually small, and users tend to keep eyes around the focuses. In this way, the number

of copy switching will be reduced, and the standstill time will be reduced and computation resource will be saved.

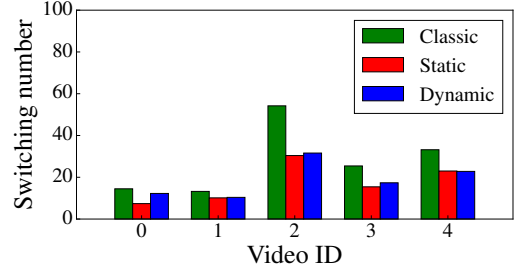


Figure 9: User ID vs. Switching Number.

**Standstill time vs. Video ID:** Our results show that *our static version can shorten standstill time by [14.9%, 53.8%], with a mean of 35.8%, and our dynamic version can shorten that by [15.3%, 40.9%], with a mean of 26.9%*, compared to the classic algorithm. As shown in Figure 10, x axis represents the bandwidth limitation and y axis represents standstill time. Note that standstill time is relative, taking naive version as benchmark. The bandwidth limitation is also relative, and we suppose the bandwidth as 1.0 with which users can watch the  $360^\circ \times 180^\circ$  high-resolution videos with no lag phases.

The bandwidth requirement of our algorithm is lower and the number of copy switching is smaller as well, thus our algorithm can significantly reduce standstill time and provide a more fluent watching experience for users.

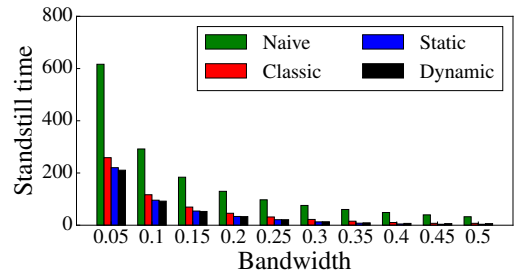


Figure 10: Bandwidth vs. Standstill time.

**High Quality Rate vs. Video ID:** Our results show that *our static version can improve high quality rate by [9.7%, 21.9%] with a mean of 16.9%, and our dynamic version can improve that by [12.1%, 19.9%] with a mean of 16.4%*, compared to the classic algorithm. As shown in Figure 11, x axis represents the video ID and y axis represents the high quality rate.

**$\alpha$  vs. Bandwidth:** Our results show that *our static version can reduce  $\alpha$  by [10.5%, 20.1%] with a mean of 15.1%, and our dynamic version can reduce that by [14.6%, 20.4%] with a mean of 18.2%*, compared to the

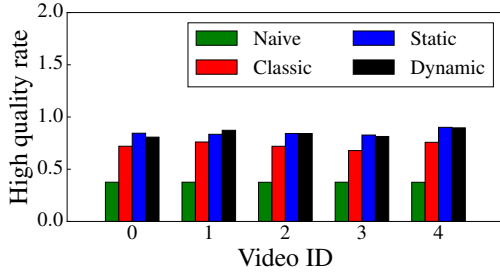


Figure 11: User ID vs. High quality rate.

classic algorithm. As shown in Figure 12, x axis represents the bandwidth and y axis represents  $\alpha$ .

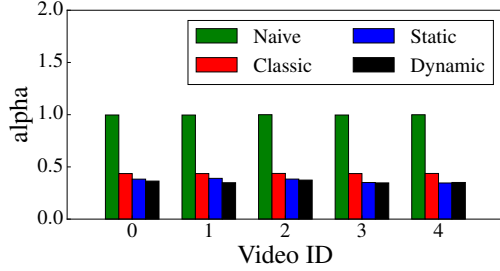


Figure 12: Video ID vs. Ratio of transmitted bytes  $\alpha$ .

## 4. CONCLUSION

Due to the requirement of low bandwidth and a superior user experience of panoramic VR videos, interactive streaming technology has drawn intensive attention in recent years. It has been put into practice by many corporations like Facebook [1], Google, Microsoft, and DWANGO Co., Ltd. [11]. However, the technology is far from mature because it brings about copy switching and degrades the user experience. Focus-based interactive streaming framework (FISF) points out a novel approach to addressing the problem by predicting behaviors of users, according to the characteristics of videos. It consists of a data-based video focus detection (VFD), two versions of FIST, and two optimizations. Experimental results show that FISF significantly improves the user experience and reduces transmission bandwidth.

To the best of our knowledge, *this is the first time* that video focus detection based on real data analysis is used to optimize panoramic VR video transmission. There are still extensive future work about FIST, like parameter choice strategy and dynamic copy producing. We believe that FISF will be implemented and widely used to provide user-friendly and bandwidth-friendly transmission for panoramic VR videos in the near future.

## 5. REFERENCES

- [1] Facebook:  
End-to-end-optimizations-for-dynamic-streaming.  
<https://code.facebook.com/posts/637561796428084/end-to-end-optimizations-for-dynamic-streaming/>.
- [2] Fisf opensource codes.  
<https://github.com/spartazhihu/FISF>.
- [3] Kandao technology co.ltd homepage.  
<http://kandaovr.com/>.
- [4] Videos used in experiments.  
[https://v1.kandaovr.com/H265/8M/zjxj-360x180\\_cube\\_lr.mp4](https://v1.kandaovr.com/H265/8M/zjxj-360x180_cube_lr.mp4),  
[https://v1.kandaovr.com/H265/8M/zjxj-360x180\\_cube\\_lr.mp4](https://v1.kandaovr.com/H265/8M/zjxj-360x180_cube_lr.mp4), <rtmp://live2.evideocloud.net/live/kandaovr>,  
[https://v1.kandaovr.com/H265/8M/guangzhou\\_4kx2k\\_cube\\_lr.mp4](https://v1.kandaovr.com/H265/8M/guangzhou_4kx2k_cube_lr.mp4),  
[https://v1.kandaovr.com/H265/8M/zebeidance2\\_360x180\\_cube\\_lr.mp4](https://v1.kandaovr.com/H265/8M/zebeidance2_360x180_cube_lr.mp4).
- [5] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010.
- [6] N. Correia and T. Chambel. Active video watching using annotation. In *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, pages 151–154. ACM, 1999.
- [7] M. Costa, N. Correia, and N. Guimarães. Annotations as multiple perspectives of video content. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 283–286. ACM, 2002.
- [8] P. d’Angelo, G. Kuschik, and P. Reinartz. Evaluation of skybox video and still image products. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(1):95, 2014.
- [9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. Density-based spatial clustering of applications with noise. In *Int. Conf. Knowledge Discovery and Data Mining*, volume 240, 1996.
- [10] Z. Huang and K. Nahrstedt. Perception-based playout scheduling for high-quality real-time interactive multimedia. In *INFOCOM, 2012 Proceedings IEEE*, pages 2786–2790. IEEE, 2012.
- [11] D. Ochi, Y. Kunita, A. Kameda, A. Kojima, and S. Iwaki. Live streaming system for omnidirectional video. In *Virtual Reality (VR), 2015 IEEE*, pages 349–350. IEEE, 2015.
- [12] I. O. Sebe, J. Hu, S. You, and U. Neumann. 3d video surveillance with augmented virtual environments. In *First ACM SIGMM international workshop on Video surveillance*, pages 107–112. ACM, 2003.
- [13] A. Vetro, H. Sun, and Y. Wang. Object-based transcoding for adaptable video content delivery. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):387–401, 2001.
- [14] O. VISION. Omnidirectional sensors for mobile robots.